

# A Pulse-Gated, Predictive Neural Circuit

Yuxiu Shao\*, Andrew T. Sornborger†, Louis Tao‡

\*Center for Bioinformatics, National Laboratory of Protein Engineering and Plant Genetic Engineering,  
College of Life Sciences, Peking University, Beijing, China  
Email: shaoyx@mail.cbi.pku.edu.cn

† Department of Mathematics, University of California, Davis, USA  
Email: ats@math.ucdavis.edu

‡Center for Bioinformatics, National Laboratory of Protein Engineering and Plant Genetic Engineering,  
College of Life Sciences, and Center for Quantitative Biology, Peking University, Beijing, China  
Email: taolt@mail.cbi.pku.edu.cn  
ATS and LT are corresponding authors.

**Abstract**—Recent evidence suggests that neural information is encoded in packets and may be flexibly routed from region to region. We have hypothesized that neural circuits are split into sub-circuits where one sub-circuit controls information propagation via pulse gating and a second sub-circuit processes graded information under the control of the first sub-circuit. Using an explicit pulse-gating mechanism, we have been able to show how information may be processed by such pulse-controlled circuits and also how, by allowing the information processing circuit to interact with the gating circuit, decisions can be made. Here, we demonstrate how Hebbian plasticity may be used to supplement our pulse-gated information processing framework by implementing a machine learning algorithm. The resulting neural circuit has a number of structures that are similar to biological neural systems, including a layered structure and information propagation driven by oscillatory gating with a complex frequency spectrum.

**Index Terms**—neural circuit, neuromorphic circuit, information gating, pulse gating, autoregressive prediction.

## I. INTRODUCTION

A considerable experimental literature indicates that 1) oscillations of various frequencies are important for the modulation of interactions in neural systems [1], [2], [3], 2) each individual pulse that makes up an oscillation may be implicated in the parallel transfer of a packet of information [4], [5], [6], [7], and 3) neural systems exist for the control of information propagation [8]. We have begun to build a neural information processing framework based on the pulse-gated propagation of graded (amplitude dependent) information [9], [10]. A key question that arises in understanding information processing in the brain is: How may neural plasticity be used to form computational circuits?

In this paper, we explore this question by creating a predictive neural circuit based on the pulse-gated control of firing rate information. We outline a set of sub-circuits responsible for sub-computations needed to estimate the process coefficients of an autoregressive (AR) learning circuit. We then combine the sub-circuits to demonstrate that a neural system can use Hebbian learning in concert with pulse-gating to predict an AR process.

## II. METHODS

### A. Autoregressive Processes

AR and moving-average (MA) processes are the two principle linear models that are used to make statistical predictions [11], [12]. AR models are filters of length  $n$  on a time series and we will consider these here. The object of this work is to implement an algorithm for predicting an AR process in a substrate of neurons. Below, we denote the filter by  $\mathbf{a}$ , the individual process covariances by  $\sigma_i$ , and the process variance-covariance matrix by  $\Sigma$ .

An AR( $n$ ) process for a zero-mean random variable,  $x_t$ , is defined

$$x_t = \sum_{i=1}^n a_i x_{t-i} + \epsilon_t.$$

To find the filter, we need to solve the Yule-Walker equations, which arise from the covariance structure of the process as follows:

$$\begin{aligned} \langle x_t x_t \rangle &= \sum_{i=1}^n a_i \langle x_{t-i} x_t \rangle + \langle \epsilon_t x_t \rangle \\ \langle x_t x_{t-1} \rangle &= \sum_{i=1}^n a_i \langle x_{t-i} x_{t-1} \rangle \\ &\vdots \\ \langle x_t x_{t-n} \rangle &= \sum_{i=1}^n a_i \langle x_{t-i} x_{t-n} \rangle \end{aligned}$$

where  $\langle \rangle$  denotes an expectation value over  $t$ . Defining  $\sigma_i = \langle x_{t-i} x_t \rangle = \langle x_t x_{t-i} \rangle$ , from the second through  $n$ 'th equations above, we have

$$\boldsymbol{\sigma} = \Sigma \mathbf{a}, \quad (1)$$

where  $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$ ,

$$\Sigma = \begin{bmatrix} \sigma_0 & \sigma_1 & \dots & \sigma_{n-1} \\ \sigma_1 & \sigma_0 & \ddots & \\ \vdots & \ddots & \ddots & \\ \sigma_{n-1} & & & \sigma_0 \end{bmatrix}, \quad (2)$$

and  $\mathbf{a} = (a_1, \dots, a_n)$ .

### B. Push-Me Pull-You Neuron Pairs

In our AR process,  $x_t$ , since we are considering an implementation in a neural circuit where firing rates encode information, the input to our system will be positive semi-definite. Both negative and positive values of  $x_t - m_0$ , where  $m_0$  is the mean, must be able to be represented by the circuit. We therefore define neuron pairs where one neuron of the pair represents positive values (i.e. the amplitude above the mean) and the other neuron represents negative values (i.e. the absolute value of the amplitude below the mean). At any given moment, only one neuron will encode non-zero amplitude in such a pair. In order to read in and encode positive and negative values of the input in a pair of neurons, we let  $m_0 = \langle x_t \rangle$ ,

$$\tau \frac{dm_1}{dt} = -m_1 + [x - m_0 + p(t) - g_0]^+$$

and

$$\tau \frac{dm_2}{dt} = -m_2 + [m_0 - x + p(t) - g_0]^+,$$

where  $p(t)$  is a square pulse of duration  $T$ . Here,  $g_0$  is a threshold value and we assume that  $|x - m_0|$  is less than the amplitude of the pulse,  $p(t)$ , so neuron 1 (firing rate  $m_1$ ) will only fire when the input is simultaneously pulsed and above the mean and, similarly, neuron 2 will only fire when the input is simultaneously pulsed and below the mean. We call such a pair of neurons a push-me pull-you (PMPY) pair. Additionally, PMPY amplitudes may be exactly propagated along a chain via pulse-gating, and we will also designate pairs along the chain as PMPY pairs.

### C. Recursive, Gradient Descent Solution to $\sigma = \Sigma \mathbf{a}$

To make predictions, we need to estimate  $\mathbf{a}$  for the process. To solve  $\sigma = \Sigma \mathbf{a}$ , we find the (unique) zero of  $\sigma - \Sigma \mathbf{a}$  using gradient descent:

$$\begin{aligned} \tau \frac{d\mathbf{a}}{dt} &= -\frac{1}{2} \frac{\delta}{\delta \mathbf{a}} (\|\sigma - \Sigma \mathbf{a}\|^2) \\ &= \Sigma (\sigma - \Sigma \mathbf{a}). \end{aligned}$$

Discretizing to first order in time gives

$$\mathbf{a}_{n+1} = \mathbf{a}_n + \frac{\Delta t}{\tau} \Sigma (\sigma - \Sigma \mathbf{a}_n), \quad (3)$$

and, since the eigenvalues of the symmetric matrix  $\Sigma$  are positive and as long as  $\Delta t/\tau$  is sufficiently small, iteration will give  $\mathbf{a}_\infty \rightarrow \Sigma^{-1} \sigma$  as  $\tau \rightarrow \infty$ .

In order to see how this solution may be computed using positive-only elements such as when performing the calculation with a set of PMPY pairs, we consider a pair of positive, semi-definite time series derived from  $x_t$ ,  $(x_t^+, x_t^-)$ , where  $x_t^+ - x_t^- = x_t$ . That is, the part of  $x_t$  that is greater than the mean in a PMPY pair is contained in  $x_t^+$ , and the absolute value of the part of  $x_t$  that is less than the mean is contained in  $x_t^-$ . The covariances associated with this time series are  $\sigma_i^{++} \equiv \langle x_{t-i}^+ x_t^+ \rangle$ ,  $\sigma_i^{+-} \equiv \langle x_{t-i}^+ x_t^- \rangle$ ,  $\sigma_i^{-+} \equiv \langle x_{t-i}^- x_t^+ \rangle$ , and

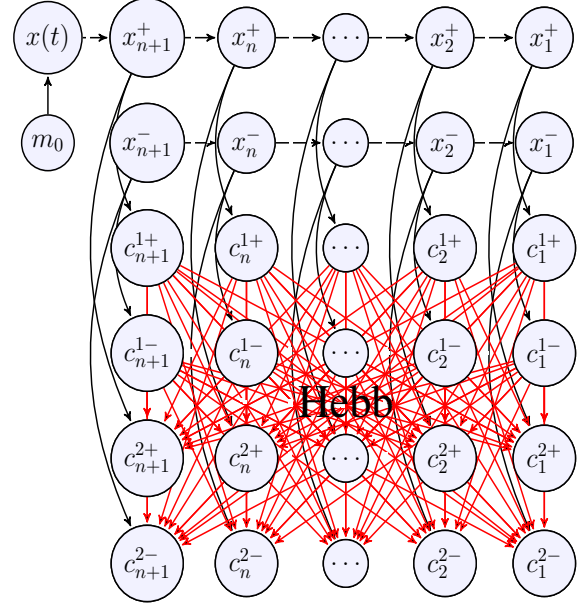


Fig. 1. The neural circuit that computes covariances,  $\sigma_i^{++}$ ,  $\sigma_i^{+-}$ , etc., between lagged values,  $x(t - T)$ , of the input time series. The mean,  $m_0$ , is first removed from the series. Then, using PMPY pairs, positive and negative values of the series are propagated for  $n+1$  lags. These same values are copied into two sets of populations among which are feedforward, all-to-all connections. Hebbian plasticity (Hebb) acts on the synapses between the two sets of populations, generating a synaptic connectivity that is an estimate of the lagged covariance.

$\sigma_i^{--} \equiv \langle x_{t-i}^- x_t^- \rangle$ . Note that  $\sigma_i = \sigma_i^{++} - \sigma_i^{+-} - \sigma_i^{-+} + \sigma_i^{--}$ . Note also that if we define

$$\begin{aligned} \Xi &= \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & \dots & \dots \\ 0 & 0 & 1 & -1 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 & -1 & \dots & \dots \end{bmatrix}, \\ \Gamma &= \begin{bmatrix} \sigma_0^{++} & \sigma_0^{+-} & \sigma_1^{++} & \sigma_1^{+-} & \dots & \dots \\ \sigma_0^{-+} & \sigma_0^{--} & \sigma_1^{-+} & \sigma_1^{--} & \dots & \dots \\ \sigma_1^{++} & \sigma_1^{+-} & \sigma_0^{++} & \sigma_0^{+-} & \dots & \dots \\ \sigma_1^{-+} & \sigma_1^{--} & \sigma_0^{-+} & \sigma_0^{--} & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{bmatrix} \quad (4) \end{aligned}$$

$\gamma = (\sigma_1^{++}, \sigma_1^{+-}, \sigma_2^{++}, \sigma_2^{+-}, \dots)$ , and  $\mathbf{p} = (a_1^+, a_1^-, a_2^+, a_2^-, \dots)$ , then  $\Xi \gamma = \sigma$ , and  $\Xi \Gamma \Xi^T = \Sigma$ . Thus, if we first compute the solution to

$$\tau \frac{d\mathbf{p}}{dt} = \Gamma (\gamma - \Gamma \mathbf{p})$$

then,  $\Xi \mathbf{p} = \mathbf{a}$ . It may be seen directly that this gradient descent projects to the one described above since

$$\begin{aligned} \tau \frac{d\mathbf{a}}{dt} &= \tau \Xi \frac{d\mathbf{p}}{dt} = \Xi \Gamma \Xi^T (\Xi \gamma - \Xi \Gamma \Xi^T \Xi \mathbf{p}) \\ &= \Sigma (\sigma - \Sigma \mathbf{a}). \end{aligned} \quad (5)$$

The reason this works is that multiplication of a vector,  $\mathbf{v}$ , by  $\Xi$  returns a new vector containing the differences of each pair of elements of  $\mathbf{v}$ , and the matrix product  $\Xi \mathbf{X} \Xi^T$  returns

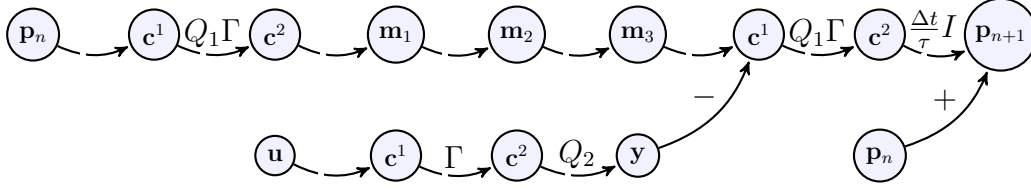


Fig. 2. The gradient descent algorithm: the coefficient estimate,  $\mathbf{p}_n$ , a vector of PMPY pairs, here represented as one node in the graph, is copied via pulse gating into  $\mathbf{c}^1$ , the vector of PMPY pairs in the middle two rows of Fig. 1, here represented as one node. This vector is then transformed via the synaptic connectivities,  $\Gamma$ , learned with Hebbian plasticity, using appropriate pulsing,  $Q_1$  acts on the output, with the result that  $Q_1 \Gamma \mathbf{p}_n$  is written into the PMPY vector  $\mathbf{c}^2$ , in the lower two rows of Fig. 1. This result is then propagated through a set of memory populations,  $\mathbf{m}_1, \dots, \mathbf{m}_3$ . After a sufficient delay such that  $\mathbf{c}^1$  has had time to decay to a firing rate of approximately 0,  $\mathbf{u}$  is read into  $\mathbf{c}^1$ , then transformed such that  $\Gamma \mathbf{u}$  is now in  $\mathbf{c}^2$ . The fixed connectivity  $Q_2$  acts on this result giving  $Q_2 \Gamma \mathbf{u}$  in  $\mathbf{y}$ . The streams are then merged with inhibitory connectivity acting to subtract  $\mathbf{m}_3$  from  $\mathbf{y}$ , such that the value  $Q_2 \Gamma \mathbf{u} - Q_1 \Gamma \mathbf{p}_n$  is in  $\mathbf{z}$ . Finally, via the last two operations,  $\mathbf{p}_{n+1}$  is set to  $\mathbf{p}_n + \frac{\Delta t}{\tau} Q_1 \Gamma (\gamma - Q_1 \Gamma \mathbf{p}_n)$ . In this way, one step of gradient descent is achieved.

a new matrix containing the difference of the sum of diagonal elements of each  $2 \times 2$  block submatrix and the sum of its off diagonal elements.

Finally, it will be useful in the discussion of the neural circuit used to implement gradient descent to extend the above considerations by using a matrix  $\Gamma'$ , which is defined as in (4), but based on an  $(n+1) \times (n+1)$  matrix  $\Sigma$ . Thus,  $\Gamma'$  will contain covariances up to a lag of  $n+1$ . In this case, we can explicitly write the gradient descent algorithm as

$$\mathbf{p}'_{n+1} = \mathbf{p}'_n + \frac{\Delta t}{\tau} Q_1 \Gamma' (\gamma' - Q_1 \Gamma' \mathbf{p}'_n), \quad (6)$$

where  $\gamma' \equiv Q_2 \Gamma' \mathbf{u}$ ,  $\mathbf{u} \equiv (1, 0, 0, 0, \dots)$ ,

$$\begin{aligned} Q_1 &= \text{diag}([1, 1, \dots, 1, 1, 0, 0]) \\ Q_2 &= \text{diag}([1, 1, \dots, 1, 1]_n, 2) \end{aligned}$$

(i.e.  $Q_2$  is a matrix of zeros, except for a 2nd superdiagonal of 1's), and

$$\mathbf{p}' = (a_1^+, a_1^-, \dots, a_n^+, a_n^-, 0, 0).$$

We have done this seemingly nonsensical extension because now  $\gamma'$  is generated by a vector input to the covariance matrix  $\Gamma'$ , as opposed to being a fixed vector. Thus, the algorithm will work for an arbitrary AR(n) time series covariance.

#### D. The Neural Circuit

Throughout our circuits, we use the pulse-gated propagation of firing rates described in [9].

1) *Computing Lagged Covariances:* The first major structure in the circuit is set up to use Hebbian plasticity to calculate covariances,  $\sigma_i^{++}$ ,  $\sigma_i^{+-}$ , etc., between successively delayed random variables in the input. To do this, the input,  $x(t)$ , is discretized and bound via a pulse into the circuit as current or firing rate amplitude packets,  $x_i^+ \equiv [x(t_i) - m_0]^+$  and  $x_i^- \equiv [m_0 - x(t_i)]^+$ , in PMPY pairs. These packets are propagated in a chain of  $n+1$  pairs. Additionally, two copies of each pair are made using a pulse gated copying procedure.

Hebbian plasticity is meant to mimic synaptic plasticity. It is a phenomenon in which coincident spiking activity between two neurons serves to increase the synaptic weight between the neurons. Since the probability of two spikes being coincident is proportional to the product of the firing rates, over time

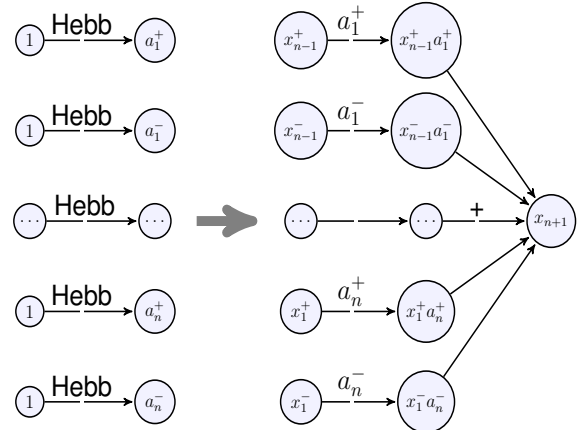


Fig. 3. Prediction: On the left, populations initialized with the value 1 are connected one-to-one with populations containing estimated coefficients. Hebbian plasticity causes the values of the estimated coefficients to be encoded in the synapses. On the right, once the synapses are sufficiently stable, lagged time series values are written into the populations previously initialized with 1 (1-populations). With a pulse, the positive components are computed, then summed, resulting in a prediction of  $x(t+1)^+$ . Initializing the 1-populations with swapped + and - components (not shown) of  $x$  results in  $x(t+1)^-$ . Estimates of future values of  $x$  may subsequently be made with the same circuit.

the synaptic strength becomes an estimate of the covariance between the firing rates of the two neurons (or in our case, neuronal populations). In our AR circuit, the populations of copied PMPY pairs interact with all-to-all connectivity (i.e. both populations in each pair in the first set of copies are connected with both populations in each pair in the second set of copies, see Fig. 1).

Synaptic weights,  $\{s_j\}$ , are set according to

$$\tau_s \frac{ds_j}{dt} = -(s_j - x(t)y(t)),$$

where  $x$  and  $y$  are firing rates from two neuronal populations. We assume that the input is in the form of a train of delta functions,  $z(t) = \sum_i z_i \delta(t - t_i)$ , that are read in by a pulse-gated neural population. After gating into a neural population, we have  $x(t) = z(t) * G(t)$ , where  $G(t)$  is the pulse envelope

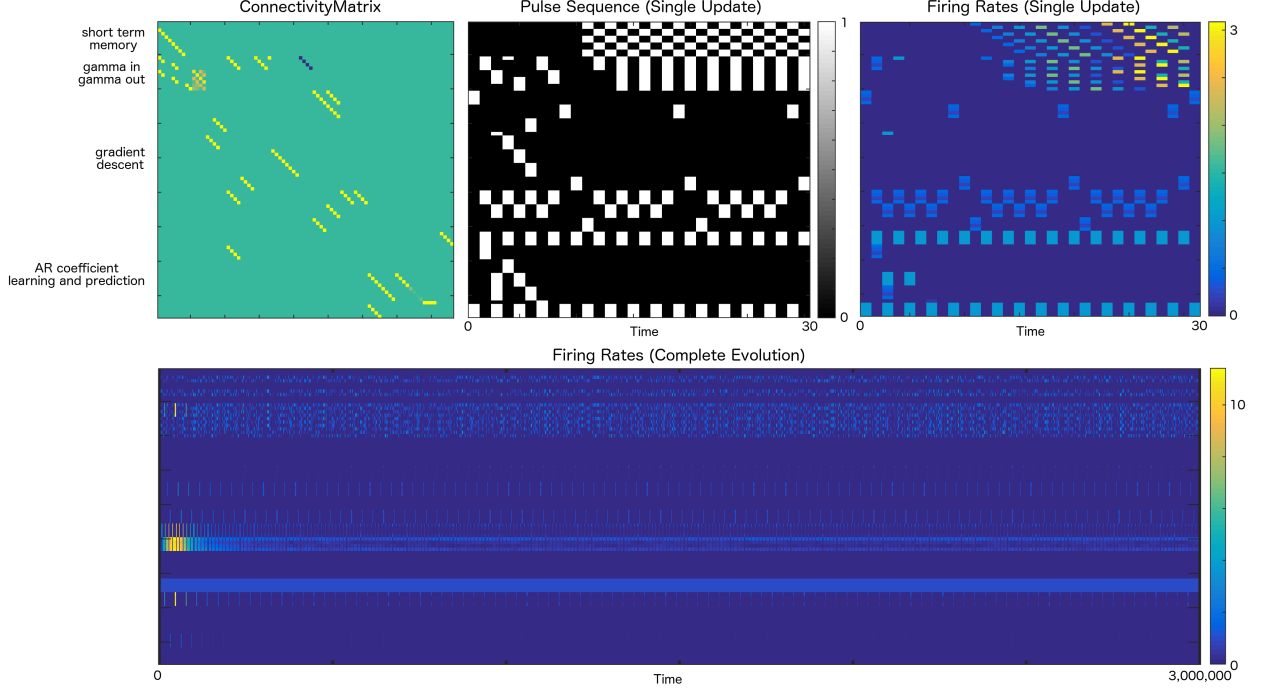


Fig. 4. The Basic Pulse-Gated Algorithm: Connectivity matrix (top left), pulse sequence (top center) and neuronal population firing rates (top right for a single update and bottom for the complete evolution). Covariance, gradient descent, and prediction circuits are activated such that they do not interfere with each other, see text for complete description.

from the gating operation,

$$G(t) = \begin{cases} \frac{t}{T} e^{-t/\tau}, & 0 < t < T \\ \frac{T}{\tau} e^{-t/\tau}, & T < t < \infty \end{cases}.$$

In the limit that the Hebbian timescale is much greater than the synaptic timescale,  $\tau_s \gg \tau$ , and the gated pulses overlap only negligibly, we have

$$s_j(t) \approx \hat{\sigma}_j,$$

where  $\hat{\sigma}_j = 4\tau/\tau_s \sum_i z_i z_{i-j}$  is an estimate of the covariance between  $x$  and  $y$  over  $\tau_s/4\tau$  samples.

Once the synaptic strengths in the copy populations have reached an equilibrium, and using pulse gating such that the synapses are feedforward between PMPY copy 1 and PMPY copy 2, we have  $\mathbf{c}^2 = \Gamma \mathbf{c}^1$ . Here, for simplicity, we have discarded the ' and just write  $\Gamma$ .

2) *Gradient Descent*: To implement the gradient descent part of the algorithm, we implement (6) using pulse gating. A method to do this is shown in Fig. 2. Here, a vector of PMPY pairs containing  $\mathbf{p}_n$  is gated into the first copy of lagged values from the time series. The synaptic connectivity, learned using Hebbian plasticity, that contains an estimate of the covariance matrix is used (along with appropriate pulses) to perform the operation  $\mathbf{c}^2 := Q_1 \Gamma \mathbf{p}_n$ . This result is then stored in a short-term memory. During this calculation, but after sufficient delay that  $\mathbf{c}^1$  may be reused,  $\mathbf{u}$  is gated into  $\mathbf{c}^1$  and, via the matrix  $\Gamma$  and another connectivity matrix that encodes  $Q_2$ ,  $Q_2 \Gamma \mathbf{u}$  is computed and stored in  $\mathbf{y}$ . The two computation streams are then combined with appropriate inhibition to subtract  $\mathbf{m}_3$  from  $\mathbf{y}$  giving  $\mathbf{z} = \gamma - Q_1 \Gamma \mathbf{p}_n$ . Two subsequent operations result in  $\mathbf{p}_{n+1}$  being set to  $\mathbf{p}_n + \frac{\Delta t}{\tau} Q_1 \Gamma (\gamma - Q_1 \Gamma \mathbf{p}_n)$

3) *Prediction*: To predict future values of  $x(t)$ , we need to form estimates of  $x(t)$  using the AR(n) process coefficients. The computation to do this requires us to compute

$$\langle x_t^+ \rangle = \sum_{i=1}^n (\langle a_i^+ x_{t-i}^+ \rangle + \langle a_i^- x_{t-i}^- \rangle)$$

and

$$\langle x_t^- \rangle = \sum_{i=1}^n (\langle a_i^+ x_{t-i}^- \rangle + \langle a_i^- x_{t-i}^+ \rangle)$$

using the coefficients,  $\mathbf{p} = (a_1^+, a_1^-, \dots)$ , that we have estimated with the gradient descent algorithm. One way that this may be done is depicted in Fig. 3.

Here, by initializing a set of populations to 1 (1-populations), connecting them one-to-one with the elements of  $\mathbf{p}_n$ , Hebbian plasticity causes the synapses to encode the values of the elements of  $\mathbf{p}$ . This is a method for creating a long-term memory. Subsequently,  $\mathbf{x}$  is copied to the 1-populations, then gated to compute  $\{\langle a_i^+ x_{t-i}^+ \rangle, \langle a_i^- x_{t-i}^- \rangle, \dots\}$ . These values are then summed to compute  $\langle x_t^+ \rangle$ . By swapping the + and - elements of  $\mathbf{x}$ , then copying to the 1-populations, the same circuit results in  $\langle x_t^- \rangle$ . The output may be stored in memory or used in other sub-circuits in the usual way.

### III. RESULTS

In Fig. 4, we depict results for the AR(2) process,  $x_{n+1} = a_1 x_n + a_2 x_{n-1} + \epsilon$ , where  $\epsilon$  is a zero mean Gaussian noise process,  $a_1 = 3/4$  and  $a_2 = -1/2$ . At the beginning of the sequence, AR input to short term memory is silenced to allow operation of the gradient descent circuit. At the same time,

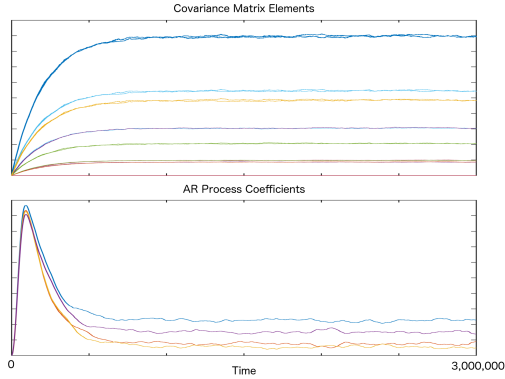


Fig. 5. Covariance and AR Coefficient Estimates: The evolution of estimates converges after approximately 30,000 updates (30 pulses per update).

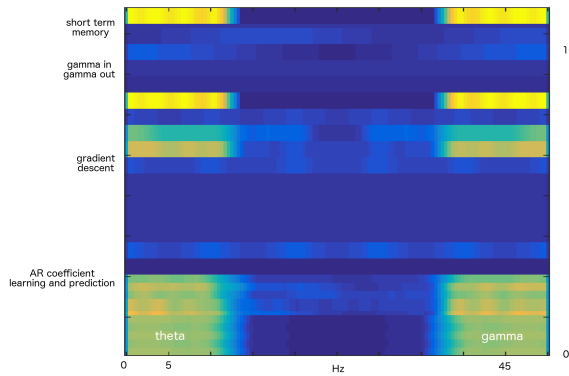


Fig. 6. Spectrum of AR Pulse Sequence: For gating pulses of length 10ms, the pulse sequence spectrum has peaks in the theta and gamma bands.

the process coefficients, encoded in firing rate amplitudes are propagated to the coefficient learning and prediction circuit to be encoded in synaptic weights. Once the gradient descent circuit and learning and prediction circuits have updated the process coefficient estimates,  $a_1$  and  $a_2$ , information needed for memory is propagated internally to each circuit and AR input is allowed to enter the covariance learning circuit. This update sequence is then repeated to obtain progressively better estimates over a long time (100,000 repeats for this simulation).

In Fig. 5, we show the convergence of the algorithm to steady state estimates for both the covariance matrix elements and AR process coefficients. Final AR process coefficients are within 6% of theoretical values. Better estimates may be obtained by increasing the learning time constant,  $\tau$ .

#### IV. CONCLUSION

Using a pulse-gating paradigm for propagating information in a neural circuit in concert with Hebbian learning, we have described how to implement an algorithm for predicting future values of an arbitrary  $AR(n)$  process. We have implemented the algorithm for an  $AR(2)$  process. The algorithm consists of three sub-circuits responsible for 1) learning the covariance matrix, 2) computing the process coefficients and 3) making a prediction by learning the process coefficients, then performing the prediction.

Using only pulse-gating, Hebbian learning and standard neuronal synaptic properties, we implemented a short-term memory, a gradient descent algorithm, a long-term memory and a method for computing an inner product to make a prediction. Additionally, the structure of the algorithm defines the brain rhythms that arise during information processing. In Fig. 6, we show the spectra of the gating pulses in the pulse sequence. For this figure, we used pulses of 10ms duration, resulting in strong gamma and theta rhythms. To our knowledge, this is the first time a neuronal model has been used to relate algorithmic structure with neuronal oscillation structure.

#### ACKNOWLEDGMENT

L.T. thanks the UC Davis Mathematics Department for its hospitality. A.T.S. would like to thank Liping Wei and the Center for Bioinformatics at the College of Life Sciences at Peking University for their hospitality. This work was supported by the Natural Science Foundation of China grant 91232715 (YXS and LT), by the Open Research Fund of the State Key Laboratory of Cognitive Neuroscience and Learning grant CNLZD1404 (YXS and LT), and by the Beijing Municipal Science and Technology Commission under contract Z151100000915070 (YXS and LT).

#### REFERENCES

- [1] R. Azouz and C. Gray, "Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons in vivo," *Proc. Natl. Acad. Sci. USA*, vol. 97, pp. 8110–8115, 2000.
- [2] P. Fries, T. Womelsdorf, R. Oostenveld, and R. Desimone, "The effects of visual stimulation and selective visual attention on rhythmic neuronal synchronization in macaque area V4," *J. Neurosci.*, vol. 28, pp. 4823–4835, 2008.
- [3] A. Markowska, D. Olton, and B. Givens, "Cholinergic manipulations in the medial septal area: Age-related effects on working memory and hippocampal electrophysiology," *J. Neurosci.*, vol. 15, pp. 2063–2073, 1995.
- [4] P. Fries, "A mechanism for cognitive dynamics: Neuronal communication through neuronal coherence," *Trends Cogn. Sci.*, vol. 9, pp. 474–480, 2005.
- [5] E. Salinas and T. Sejnowski, "Impact of correlated synaptic input on output firing rate and variability in simple neuronal models," *J. Neurosci.*, vol. 20, pp. 6193–6209, 2000.
- [6] A. Luczak, P. Bartho, S. L. Marguet, G. Buzsaki, and K. D. Harris, "Sequential structure of neocortical spontaneous activity in vivo," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 104, pp. 347–352, 2007.
- [7] A. Luczak, B. L. McNaughton, and K. D. Harris, "Packet-based communication in the cortex," *Nat. Rev. Neurosci.*, vol. 16, no. 12, pp. 745–755, Dec 2015.
- [8] T. Gisiger and M. Boukadoum, "Mechanisms Gating the Flow of Information in the Cortex: What They Might Look Like and What Their Uses may be," *Front Comput Neurosci*, vol. 5, p. 1, 2011.
- [9] A. Sornborger, Z. Wang, and L. Tao, "A mechanism for graded, dynamically routable current propagation in pulse-gated synfire chains and implications for information coding," *J. Comput. Neurosci.*, vol. 39, pp. 181–195, August 2015.
- [10] Z. Wang, A. T. Sornborger, and L. Tao, "Graded, Dynamically Routable Information Processing with Synfire-Gated Synfire Chains," *PLoS Comput. Biol.*, vol. 12, no. 6, p. e1004979, Jun 2016.
- [11] S. Pandit and S.-M. Wu, Eds., *Time Series and System Analysis with Applications*. New York: John Wiley & Sons, 1983.
- [12] D. Percival and A. Walden, Eds., *Spectral Analysis for Physical Applications*. Cambridge, UK: Cambridge University Press, 1993.